PL/SQL Lab Report
LAB – 9

Student Name : AHNAF SHAHRIAR PIAS

Student ID : 220042146

Task 1.1: Display "Hello World"

Code:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('Hello World');
END;
/
```

Output:

```
Hello World
```

Explanation:

This task demonstrated the basic structure of a PL/SQL block and the use of `DBMS_OUTPUT.PUT_LINE` to print messages.

Task 1.2: Declare and Display Variables

Code:

```
DECLARE
    my_name VARCHAR2(50) := 'Ahnaf';
    my_id NUMBER := 101;
    friend_name VARCHAR2(50) := 'Pias';
    friend_id NUMBER := 102;
BEGIN
    DBMS_OUTPUT.PUT_LINE('My Name    :' || my_name || ', ID: ' ||
my_id);
    DBMS_OUTPUT.PUT_LINE('Friend Name: ' || friend_name || ', ID: ' ||
friend_id);
END;
/
```

Output:

```
My Name    : Ahnaf, ID: 101
Friend Name: Pias, ID: 102
```

Explanation:

This task introduced variable declaration and string concatenation using the `||` operator.

Task 1.3: Perform Arithmetic Operations

Code:

```
DECLARE
    num1 NUMBER := 10;
    num2 NUMBER := 5;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Addition       : ' || (num1 + num2));
    DBMS_OUTPUT.PUT_LINE('Subtraction    : ' || (num1 - num2));
    DBMS_OUTPUT.PUT_LINE('Multiplication: ' || (num1 * num2));
    DBMS_OUTPUT.PUT_LINE('Division       : ' || (num1 / num2));
END;
/
```

Output:

```
Addition       : 15
Subtraction    : 5
Multiplication: 50
Division       : 2
```

Explanation:

The program demonstrated mathematical operations using PL/SQL variables.

Task 1.4: Determine Grade Based on Percentage

Code:

```
DECLARE
    mark_percentage NUMBER := 85;
    grade VARCHAR2(10);
BEGIN
    IF mark_percentage >= 90 THEN
        grade := 'A';
    ELSIF mark_percentage >= 80 THEN
        grade := 'B';
    ELSIF mark_percentage >= 70 THEN
        grade := 'C';
    ELSIF mark_percentage >= 60 THEN
        grade := 'D';
    ELSE
        grade := 'F';
    END IF;
    DBMS_OUTPUT.PUT_LINE('Grade: ' || grade);
END;
/
```

Output:

```
Grade: B
```

Explanation:

This task illustrated the use of `IF-ELSE` conditions for decision-making.

Task 1.5: Print 68 Student IDs Using a While Loop

Code:

```
DECLARE
    student_id NUMBER := 101;
    counter NUMBER := 1;
BEGIN
    WHILE counter <= 68 LOOP
        DBMS_OUTPUT.PUT_LINE('Student ID: ' || student_id);
        student_id := student_id + 1;
        counter := counter + 1;
    END LOOP;
END;
/
```

Output:

```
Student ID: 101
Student ID: 102
...
Student ID: 168
```

Explanation:

This task demonstrated the use of loops to execute repetitive tasks efficiently.

Task 2: Determine Decade Start Year

Code:

```
DECLARE
    current_year NUMBER := 2024;
    decade_start NUMBER;
BEGIN
    IF MOD(current_year, 10) = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Yes');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No');
    END IF;
    decade_start := current_year - MOD(current_year, 10);
    DBMS_OUTPUT.PUT_LINE('The ' || decade_start || 's');
END;
/
```

Output:

```
No
The 2020s
```

Explanation:

This task introduced modular arithmetic and string concatenation.

Task 3: Sum of Prime Numbers Until 20

Code:

```
DECLARE
    num NUMBER := 2;
    sum_primes NUMBER := 0;
    is_prime BOOLEAN;
    i NUMBER;
BEGIN
    WHILE sum_primes + num <= 20 LOOP
        is_prime := TRUE;
        FOR i IN 2..TRUNC(SQRT(num)) LOOP
            IF MOD(num, i) = 0 THEN
                is_prime := FALSE;
                EXIT;
            END IF;
        END LOOP;

        IF is_prime THEN
            DBMS_OUTPUT.PUT_LINE(num);
            sum_primes := sum_primes + num;
        END IF;

        num := num + 1;
    END LOOP;
END;
/
```

Output:

```
2
3
5
7
```

Explanation:

This task involved loop control structures and prime number identification using the square root method.
